Bhavin Patel

COMP 301
PROJECT 1
WORKING ALONE

Step 1 – Create a public/private key pair for the TTP

Commands:

gpg –full-generate-key
1
4096
6m
Y
Real Name: userTTP
Email: userttp@ttp.com
o

```
bpatel13@comp301-f21:~$ gpg --full-generate-key
gpg (GnuPG) 2.2.19; Copyright (C) 2019 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
  (14) Existing key from card
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072) 4096
Requested keysize is 4096 bits
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0) 6m
Key expires at Tue 29 Mar 2022 11:29:18 AM CDT
Is this correct? (y/N) y

GnuPG needs to construct a user ID to identify your key.

Real name: userTTP
Email address: userttp@ttp.com
Comment:
You selected this USER-ID:
    "userTTP <userttp@ttp.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: key 731197D10F7DB45D marked as ultimately trusted
gpg: revocation certificate stored as '/home/bpatel13/.gnupg/openpgp-revocs.d/95D8118F3DCDDA9FDBA45197731197D10F7DB45D.rev'
public and secret key created and signed.

pub   rsa4096 2021-09-30 [SC] [expires: 2022-03-29]
      95D8118F3DCDDA9FDBA45197731197D10F7DB45D
uid                      userTTP <userttp@ttp.com>
sub   rsa4096 2021-09-30 [E] [expires: 2022-03-29]

bpatel13@comp301-f21:~$
```

Step 2 – Import public key of TTP for C

Commands:

gpg –export –a userttp@ttp.com > client.pub
cp client.pub /usr/local/src/project1-bpatel13
gpg –import /usr/local/src/project1-bpatel13/client.pub
gpg –list-public-keys

```
bpatel13@comp301-f21:~$ gpg --export -a userttp@ttp.com > client.pub
bpatel13@comp301-f21:~$ dir
bpatel13.pub  client.pub  lab2
bpatel13@comp301-f21:~$ sudo cp client.pub /usr/local/src/project1-bpatel13
[sudo] password for bpatel13:
bpatel13 is not in the sudoers file.  This incident will be reported.
bpatel13@comp301-f21:~$ cp client.pub /usr/local/src/project1-bpatel13
bpatel13@comp301-f21:~$ gpg --import /usr/local/src/project1-bpatel13/client.pub
gpg: key 731197D10F7DB45D: "userTTP <userttp@ttp.com>" not changed
gpg: Total number processed: 1
gpg:              unchanged: 1
bpatel13@comp301-f21:~$ gpg --list-public-keys
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   2  signed:   1  trust: 0-, 0q, 0n, 0m, 0f, 2u
gpg: depth: 1  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 1f, 0u
gpg: next trustdb check due at 2022-03-03
/home/bpatel13/.gnupg/pubring.kbx
---------------------------------
pub   rsa4096 2021-09-27 [SC] [expires: 2022-03-26]
      20A3653B1668ADFE91C6FACFFC397E4CBF5F85BB
uid           [ultimate] Bhavin Patel <bpatel13@luc.edu>
sub   rsa4096 2021-09-27 [E] [expires: 2022-03-26]

pub   rsa4096 2021-09-04 [SC] [expires: 2022-03-03]
      88B656F63407C421BC1FDD6E7C8A52B36B92841A
uid           [  full  ] Eric Chan-Tin <chantin@cs.luc.edu>
sub   rsa4096 2021-09-04 [E] [expires: 2022-03-03]

pub   rsa4096 2021-09-30 [SC] [expires: 2022-03-29]
      95D8118F3DCDDA9FDBA45197731197D10F7DB45D
uid           [ultimate] userTTP <userttp@ttp.com>
sub   rsa4096 2021-09-30 [E] [expires: 2022-03-29]

bpatel13@comp301-f21:~$
```

Step 3 – Create public/private key pair for S

Commands:

gpg –full-generate-key
1
4096
6m
Y
Real name: userS
Email: userS.s.com
gpg –export –a userS@s.com > server.pub
cp server.pub /usr/local/src/project1-bpatel132

```
bpatel132@comp301-f21:~$ gpg --full-generate-key
gpg (GnuPG) 2.2.19; Copyright (C) 2019 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

gpg: directory '/home/bpatel132/.gnupg' created
gpg: keybox '/home/bpatel132/.gnupg/pubring.kbx' created
Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
  (14) Existing key from card
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072) 4096
Requested keysize is 4096 bits
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0) 6m
Key expires at Tue 29 Mar 2022 12:28:06 PM CDT
Is this correct? (y/N) y

GnuPG needs to construct a user ID to identify your key.

Real name: userS
Email address: userS@s.com
Comment:
You selected this USER-ID:
    "userS <userS@s.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: /home/bpatel132/.gnupg/trustdb.gpg: trustdb created
gpg: key 1B92FAAAB18B40FD marked as ultimately trusted
gpg: directory '/home/bpatel132/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/home/bpatel132/.gnupg/openpgp-revocs.d/92251EF22B8ADA288D50B28E1B92FAAAB18B40FD.rev'
public and secret key created and signed.

pub   rsa4096 2021-09-30 [SC] [expires: 2022-03-29]
      92251EF22B8ADA288D50B28E1B92FAAAB18B40FD
uid                      userS <userS@s.com>
sub   rsa4096 2021-09-30 [E] [expires: 2022-03-29]

bpatel132@comp301-f21:~$
bpatel132@comp301-f21:~$ gpg --export -a userS@s.com > server.pub
bpatel132@comp301-f21:~$ cp server.pub /usr/local/src/project1-bpatel132
bpatel132@comp301-f21:~$
```

Step 4 – Have TTP sign the public key for S

Commands:

gpg –import /usr/local/src/project1-bpatel32/server.pub
gpg –u 95D8118F3DCDDA9FDBA45197731197D10F7DB45D –edit-key
trust
4
lsign
y
save

Bhavin Patel COMP 301
PROJECT 1
WORKING ALONE

```
bpatel13@comp301-f21:~$ gpg --import /usr/local/src/project1-bpatel132/server.pub
gpg: key 1B92FAAAB18B40FD: "userS <userS@s.com>" not changed
gpg: Total number processed: 1
gpg:              unchanged: 1
bpatel13@comp301-f21:~$ gpg -u 95D8118F3DCDDA9FDBA45197731197D10F7DB45D --edit-key
gpg (GnuPG) 2.2.19; Copyright (C) 2019 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.


pub  rsa4096/1B92FAAAB18B40FD
     created: 2021-09-30  expires: 2022-03-29  usage: SC
     trust: unknown       validity: unknown
sub  rsa4096/4CB3426A8B2A4C8C
     created: 2021-09-30  expires: 2022-03-29  usage: E
[ unknown] (1). userS <userS@s.com>

gpg> trust
pub  rsa4096/1B92FAAAB18B40FD
     created: 2021-09-30  expires: 2022-03-29  usage: SC
     trust: unknown       validity: unknown
sub  rsa4096/4CB3426A8B2A4C8C
     created: 2021-09-30  expires: 2022-03-29  usage: E
[ unknown] (1). userS <userS@s.com>

Please decide how far you trust this user to correctly verify other users' keys
(by looking at passports, checking fingerprints from different sources, etc.)

  1 = I don't know or won't say
  2 = I do NOT trust
  3 = I trust marginally
  4 = I trust fully
  5 = I trust ultimately
  m = back to the main menu

Your decision? 4

pub  rsa4096/1B92FAAAB18B40FD
     created: 2021-09-30  expires: 2022-03-29  usage: SC
     trust: full          validity: unknown
sub  rsa4096/4CB3426A8B2A4C8C
     created: 2021-09-30  expires: 2022-03-29  usage: E
[ unknown] (1). userS <userS@s.com>
Please note that the shown key validity is not necessarily correct
unless you restart the program.

gpg> lsign

pub  rsa4096/1B92FAAAB18B40FD
     created: 2021-09-30  expires: 2022-03-29  usage: SC
     trust: full          validity: unknown
 Primary key fingerprint: 9225 1EF2 2B8A DA28 8D50  B28E 1B92 FAAA B18B 40FD

     userS <userS@s.com>

This key is due to expire on 2022-03-29.
```

```
Are you sure that you want to sign this key with your
key "userTTP <userttp@ttp.com>" (731197D10F7DB45D)

The signature will be marked as non-exportable.

Really sign? (y/N) y

gpg> save
bpatel13@comp301-f21:~$
```

Step 5 – C "downloads" signed public key of S

Commands:

gpg –import /usr/local/src/project1-bpatel132/server.pub

```
bpatel13@comp301-f21:~$ gpg --import /usr/local/src/project1-bpatel132/server.pub
gpg: key 1B92FAAAB18B40FD: "userS <userS@s.com>" not changed
gpg: Total number processed: 1
gpg:              unchanged: 1
bpatel13@comp301-f21:~$
```

Step 6 – Verify the public key's signature is correct

Commands:

gpg –fingerprint userS@s.com
gpg –list-keys

```
bpatel13@comp301-f21:~$ gpg --fingerprint userS@s.com
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   2  signed:   2  trust: 0-, 0q, 0n, 0m, 0f, 2u
gpg: depth: 1  valid:   2  signed:   0  trust: 0-, 0q, 0n, 0m, 2f, 0u
gpg: next trustdb check due at 2022-03-03
pub   rsa4096 2021-09-30 [SC] [expires: 2022-03-29]
      9225 1EF2 2B8A DA28 8D50  B28E 1B92 FAAA B18B 40FD
uid           [  full  ] userS <userS@s.com>
sub   rsa4096 2021-09-30 [E] [expires: 2022-03-29]

bpatel13@comp301-f21:~$ gpg --list-keys
/home/bpatel13/.gnupg/pubring.kbx
---------------------------------
pub   rsa4096 2021-09-27 [SC] [expires: 2022-03-26]
      20A3653B1668ADFE91C6FACFFC397E4CBF5F85BB
uid           [ultimate] Bhavin Patel <bpatel13@luc.edu>
sub   rsa4096 2021-09-27 [E] [expires: 2022-03-26]

pub   rsa4096 2021-09-04 [SC] [expires: 2022-03-03]
      88B656F63407C421BC1FDD6E7C8A52B36B92841A
uid           [  full  ] Eric Chan-Tin <chantin@cs.luc.edu>
sub   rsa4096 2021-09-04 [E] [expires: 2022-03-03]

pub   rsa4096 2021-09-30 [SC] [expires: 2022-03-29]
      95D8118F3DCDDA9FDBA45197731197D10F7DB45D
uid           [ultimate] userTTP <userttp@ttp.com>
sub   rsa4096 2021-09-30 [E] [expires: 2022-03-29]

pub   rsa4096 2021-09-30 [SC] [expires: 2022-03-29]
      92251EF22B8ADA288D50B28E1B92FAAAB18B40FD
uid           [  full  ] userS <userS@s.com>
sub   rsa4096 2021-09-30 [E] [expires: 2022-03-29]

bpatel13@comp301-f21:~$
```

Step 7 – C encrypts “Hello” using S’s public key

Commands:

echo “Hello” > msg.txt
gpg –output msg.txt.gpg –encrypt –recipient userS@s.com msg.txt
cp msg.txt.gpg /usr/local/src/project1-bpatel13

```
bpatel13@comp301-f21:~$ echo "Hello" > msg.txt
bpatel13@comp301-f21:~$ gpg --output msg.txt.gpg --encrypt --recipient userS@s.com msg.txt
bpatel13@comp301-f21:~$ dir
bpatel13.pub  client.pub  lab2  msg.txt  msg.txt.gpg
bpatel13@comp301-f21:~$ cp msg.txt.gpg /usr/local/src/project1-bpatel13
bpatel13@comp301-f21:~$
```

Step 8 – S decrypts the message and sends back a list of supported algorithms

Commands:

gpg –output decr.txt –decrypt /usr/local/src/project1-bpatel13/msg.txt.gpg
cat decr.txt
openssl list –cipher-algorithms

```
bpatel132@comp301-f21:~$ gpg --output decr.txt --decrypt /usr/local/src/project1-bpatel13/msg.txt.gpg
gpg: encrypted with 4096-bit RSA key, ID 4CB3426A8B2A4C8C, created 2021-09-30
      "userS <userS@s.com>"
bpatel132@comp301-f21:~$ dir
decr.txt  server.pub
bpatel132@comp301-f21:~$ cat decr.txt
Hello
bpatel132@comp301-f21:~$
```

```
bpatel132@comp301-f21:~$ openssl list -cipher-algorithms
AES-128-CBC
AES-128-CBC-HMAC-SHA1
AES-128-CBC-HMAC-SHA256
id-aes128-CCM
AES-128-CFB
AES-128-CFB1
AES-128-CFB8
AES-128-CTR
AES-128-ECB
id-aes128-GCM
AES-128-OCB
AES-128-OFB
AES-128-XTS
AES-192-CBC
id-aes192-CCM
AES-192-CFB
AES-192-CFB1
AES-192-CFB8
AES-192-CTR
AES-192-ECB
id-aes192-GCM
AES-192-OCB
AES-192-OFB
AES-256-CBC
AES-256-CBC-HMAC-SHA1
AES-256-CBC-HMAC-SHA256
id-aes256-CCM
AES-256-CFB
AES-256-CFB1
AES-256-CFB8
AES-256-CTR
AES-256-ECB
id-aes256-GCM
AES-256-OCB
AES-256-OFB
AES-256-XTS
aes128 => AES-128-CBC
aes128-wrap => id-aes128-wrap
aes192 => AES-192-CBC
aes192-wrap => id-aes192-wrap
aes256 => AES-256-CBC
aes256-wrap => id-aes256-wrap
```

Step 9 – C selects one of these algorithms, generates a secret key, and sends that secret key and selected algorithm, encrypted under S's public key

Commands:

gpg –edit-key userTTP@ttp.com
addkey
6
6m
Y
y
save
gpg --export –a userttp@ttp.com > userC.pub
gpg –output userC.gpg –encrypt –cipher-algo aes128 –recipient userS@s.com userC.pub
cp userC.gpg /usr/local/src/project1-bpatel13

```
bpatel13@comp301-f21:~$ gpg --edit-key userTTP@ttp.com
gpg (GnuPG) 2.2.19; Copyright (C) 2019 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Secret key is available.

sec  rsa4096/731197D10F7DB45D
     created: 2021-09-30  expires: 2022-03-29  usage: SC
     trust: ultimate      validity: ultimate
ssb  rsa4096/D93540F5440217A4
     created: 2021-09-30  expires: 2022-03-29  usage: E
[ultimate] (1). userTTP <userttp@ttp.com>

gpg> addkey
Please select what kind of key you want:
   (3) DSA (sign only)
   (4) RSA (sign only)
   (5) Elgamal (encrypt only)
   (6) RSA (encrypt only)
  (14) Existing key from card
Your selection? 6
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072) 4096
Requested keysize is 4096 bits
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0) 6m
Key expires at Tue 29 Mar 2022 02:10:00 PM CDT
Is this correct? (y/N) y
Really create? (y/N) y
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.

sec  rsa4096/731197D10F7DB45D
     created: 2021-09-30  expires: 2022-03-29  usage: SC
     trust: ultimate      validity: ultimate
ssb  rsa4096/D93540F5440217A4
     created: 2021-09-30  expires: 2022-03-29  usage: E
ssb  rsa4096/0716E87B5CCD6A40
     created: 2021-09-30  expires: 2022-03-29  usage: E
[ultimate] (1). userTTP <userttp@ttp.com>

gpg> save
bpatel13@comp301-f21:~$ gpg --export -a userttp@ttp.com > userC.pub
bpatel13@comp301-f21:~$ dir
bpatel13.pub  client.pub  lab2  msg.txt  msg.txt.gpg  userC.pub
bpatel13@comp301-f21:~$
bpatel13@comp301-f21:~$  gpg --output userC.gpg --encrypt --cipher-algo aes128 --recipient userS@s.com userC.pub
bpatel13@comp301-f21:~$ dir
bpatel13.pub  client.pub  lab2  msg.txt  msg.txt.gpg  userC.gpg  userC.pub
bpatel13@comp301-f21:~$ cp userC.gpg /usr/local/src/project1-bpatel13
bpatel13@comp301-f21:~$
```

Step 10 – C encrypts “Ready” using the secret key and creates a hash

Commands:

echo “Ready” > step10.txt
gpg –output step10.txt.gpg –encrypt –cipher-algo aes128 –recipient userS@s.com step10.txt
openssl dgst –md5 step10.txt.gpg > hash.txt

```
bpatel13@comp301-f21:~$ echo "Ready" > step10.txt
bpatel13@comp301-f21:~$ dir
bpatel13.pub  client.pub  lab2  msg.txt  msg.txt.gpg  step10.txt  userC.gpg  userC.pub
bpatel13@comp301-f21:~$ gpg --output step10.txt.gpg --encrypt --cipher-algo aes128 --recipient userS@s.com step10.txt
```

```
bpatel13@comp301-f21:~$ openssl dgst -md5 step10.txt.gpg > hash.txt
bpatel13@comp301-f21:~$
```

Step 11 - C sends both the encrypted message and hash to S

Commands:

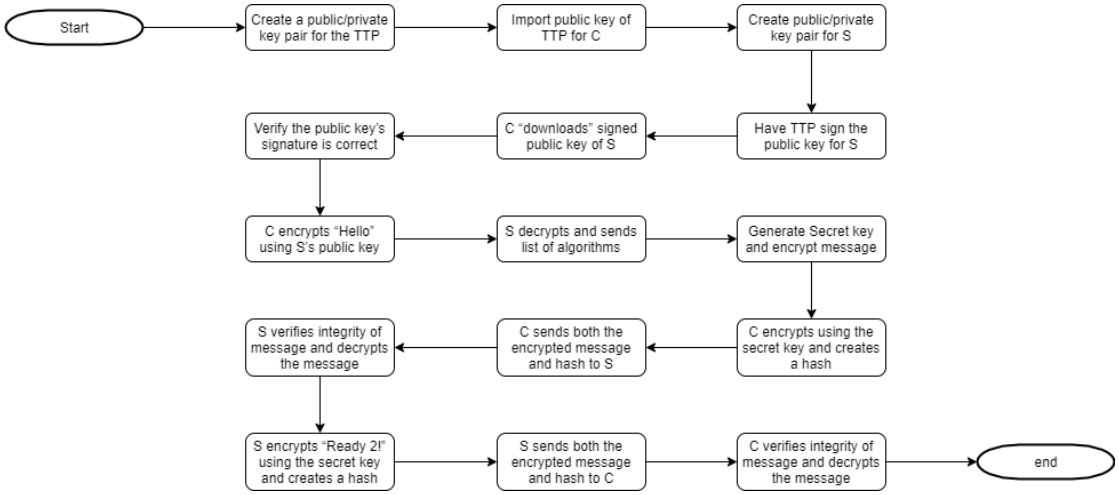
cp step10.txt.gpg /usr/local/src/project1-bpatel13
cp hash.txt /sur/local/src/project1-bpatel13

```
bpatel13@comp301-f21:~$ cp step10.txt.gpg /usr/local/src/project1-bpatel13
bpatel13@comp301-f21:~$ cp hash.txt /usr/local/src/project1-bpatel13
bpatel13@comp301-f21:~$
```

Bhavin Patel

COMP 301
PROJECT 1
WORKING ALONE

FLOWCART



Step 12 - S verifies integrity of message and decrypts the message

Commands:

cp /usr/local/src/project1-bpatel13/userC.gpg .
gpg userC.gpg
gpg –import userC
openssl dgst –md5 step10.txt.gpg
cat hash.txt
cat dcstep10.txt

```
bpatel132@comp301-f21:~$ cp /usr/local/src/project1-bpatel13/userC.gpg .
bpatel132@comp301-f21:~$ dir
decr.txt  dstep10.txt  hash.txt  server.pub  step10.txt.gpg  userC.gpg
bpatel132@comp301-f21:~$ gpg userC.gpg
gpg: WARNING: no command supplied.  Trying to guess what you mean ...
gpg: encrypted with 4096-bit RSA key, ID 4CB3426A8B2A4C8C, created 2021-09-30
      "userS <userS@s.com>"
bpatel132@comp301-f21:~$ dir
decr.txt  dstep10.txt  hash.txt  server.pub  step10.txt.gpg  userC  userC.gpg
bpatel132@comp301-f21:~$ gpg --import userC
gpg: key 731197D10F7DB45D: public key "userTTP <userttp@ttp.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1
```

```
bpatel132@comp301-f21:~$ openssl dgst -md5 step10.txt.gpg
MD5(step10.txt.gpg)= 092494219f583ff1aac44a4bb63a8fec
bpatel132@comp301-f21:~$ cat hash.txt
MD5(step10.txt.gpg)= 092494219f583ff1aac44a4bb63a8fec
bpatel132@comp301-f21:~$
bpatel132@comp301-f21:~$ gpg --output dcstep10.txt --decrypt step10.txt.gpg
gpg: encrypted with 4096-bit RSA key, ID 4CB3426A8B2A4C8C, created 2021-09-30
      "userS <userS@s.com>"
bpatel132@comp301-f21:~$ cat dcstep10.txt
Ready
bpatel132@comp301-f21:~$
```

Step 13 - S encrypts “Ready 2!” using the secret key and creates a hash

Commands:

echo “Ready 2!” > step13.txt
gpg –output step13.txt.gpg –encrypt –cipher-algo aes128 –recipient userS@s.com step13.txt
openssl dgst –md5 step13.txt > step13hash.txt
openssl dgst –md5 step13.txt.gpg > step13hash2.txt

```
bpatel132@comp301-f21:~$ echo "Ready 2!" > step13.txt
bpatel132@comp301-f21:~$ gpg --output step13.txt.gpg --encrypt --cipher-algo aes128 --recipient userS@s.com step13.txt
bpatel132@comp301-f21:~$ dir
dcstep10.txt  decr.txt  dstep10.txt  hash.txt  server.pub  step10.txt.gpg  step13.txt  step13.txt.gpg  userC  userC.gpg
bpatel132@comp301-f21:~$ openssl dgst -md5 step13.txt > step13hash.txt
bpatel132@comp301-f21:~$ openssl dgst -md5 step13.txt.gpg > step13hash2.txt
```

Step 14 - S sends both the encrypted message and hash to C

Commands:

cp step13.txt.gpg /usr/local/src/project1-bpatel132
cp step13hash.txt /usr/local/src/project1-bpatel132
cp step13hash2.txt /usr/local/src/project1-bpatel132

```
bpatel132@comp301-f21:~$ cp step13.txt.gpg /usr/local/src/project1-bpatel132
bpatel132@comp301-f21:~$ cp step13hash.txt /usr/local/src/project1-bpatel132
bpatel132@comp301-f21:~$ cp step13hash2.txt /usr/local/src/project1-bpatel132
bpatel132@comp301-f21:~$
```

Step 15 - C verifies integrity of message and decrypts the message

Commands:

cp /usr/local/src/project1-bpatel132/step13hash.txt .
cp /usr/local/src/project1-bpatel132/step13hash2.txt .
cp /usr/local/src/project1-bpatel132/step13.txt.gpg
gpg –output dcstep13.txt –decrypt step13.txt.gpg
cat dcstep13.txt
openssl dgst –md5 dcstep13.txt
openssl dgst –md5 step13.txt.gpg
cat step13hash.txt
cat step13hash2.txt

```
bpatel13@comp301-f21:~$ cp /usr/local/src/project1-bpatel132/step13hash.txt .
bpatel13@comp301-f21:~$ cp /usr/local/src/project1-bpatel132/step13hash2.txt .
bpatel13@comp301-f21:~$ cp /usr/local/src/project1-bpatel132/step13.txt.gpg .
bpatel13@comp301-f21:~$ dir
bpatel13.pub  dcstep13.txt  lab2     msg.txt.gpg  step10.txt.gpg   step13hash.txt  step13v2.txt.gpg  userC.pub
client.pub    hash.txt      msg.txt  step10.txt   step13hash2.txt  step13.txt.gpg  userC.gpg
bpatel13@comp301-f21:~$ gpg --output dcstep13.txt --decrypt step13.txt.gpg
gpg: encrypted with 4096-bit RSA key, ID 0716E87B5CCD6A40, created 2021-09-30
      "userTTP <userttp@ttp.com>"
bpatel13@comp301-f21:~$
```

```
bpatel13@comp301-f21:~$ cat dcstep13.txt
Ready 2!
bpatel13@comp301-f21:~$ openssl dgst -md5 dcstep13.txt
MD5(dcstep13.txt)= f3a2fd11c120f892695093a988469531
bpatel13@comp301-f21:~$ openssl dgst -md5 step13.txt.gpg
MD5(step13.txt.gpg)= 5af10bbf35eeb6de7f1b23fb8173d51c
bpatel13@comp301-f21:~$ cat step13hash.txt
MD5(step13.txt)= f3a2fd11c120f892695093a988469531
bpatel13@comp301-f21:~$ cat step13hash2.txt
MD5(step13.txt.gpg)= 5af10bbf35eeb6de7f1b23fb8173d51c
bpatel13@comp301-f21:~$
```